Secure User Management With JWT Implementation

By: ROHIT YADAV

Overview

Introduction

The Secure User Management System project is designed to handle user authentication, authorization, and profile management securely. The project implements JSON Web Tokens (JWT) for authentication, role-based access control (RBAC) with admin and user roles, and scopes for fine-grained permissions such as UPDATE, DELETE, GET, and POST operations. Security measures include rate limiting to prevent Dos attack or traffic, custom validation for data integrity, and password hashing using scrypt for secure storage. The project also includes functionalities for generating, validating, refreshing, and revoking JWT tokens, ensuring robust security throughout the user management process.

Hashing Method - Scrypt

Scrypt is a method used for password hashing, similar to bcrypt. However, Scrypt incorporates a key advantage by including a "salt" parameter directly within the hashing process. This salt, which is a random data input, helps prevent rainbow table attacks by ensuring that each password hash is unique, even if the passwords are identical. This internal handling of salts in Scrypt enhances security and adds an extra layer of protection against password cracking attempts.

JWT- TOKEN

01

JSON Web Tokens (JWT) are a compact and secure way of transmitting information between parties as a JSON object. They are primarily used for authorization in web applications.

02

JWTs consist of three parts: header, payload, and signature, which are base64-encoded and concatenated with periods. The header contains the token type and hashing algorithm, the payload contains the actual data (claims), and the signature is used to verify the token's authenticity.

03

JWTs are stateless tokens, meaning the server does not need to maintain session data for each user. This reduces the server's workload and makes it easier to scale the application. Unlike traditional session-based authentication methods, which require the server to store session data (like session IDs) for each user.

API

SignUp ->

Login ->

GetDetails ->

DeleteUser ->

UpdateUser ->

ChangePassword ->

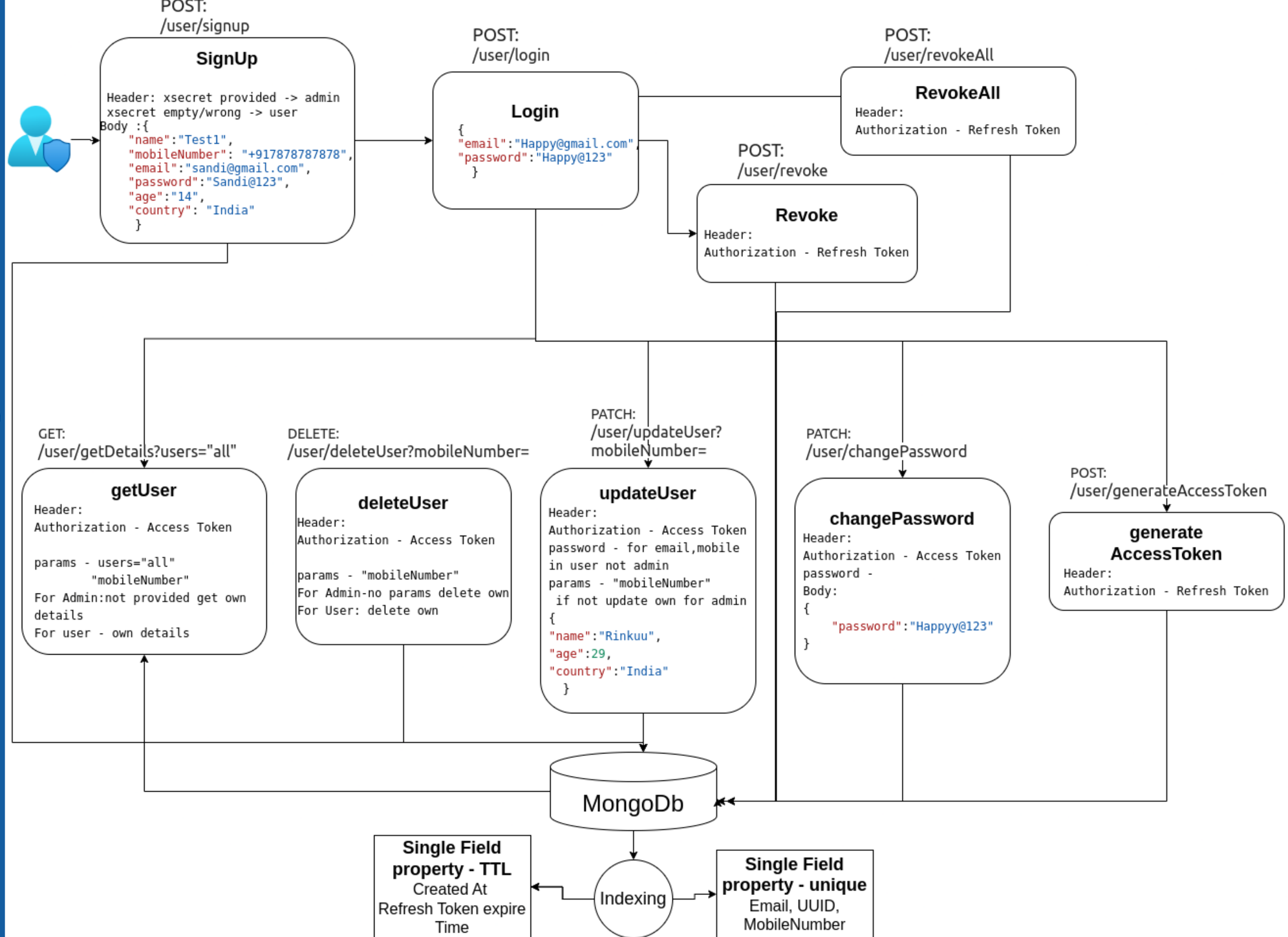
GenerateAccessToken ->

Revoke ->

RevokeAll ->

API	Type	Explanation
http://localhost:3000/user/signup	POST	signUp the user and registers him in database
http://localhost:3000/user/login	POST	login the user and gives him access token and refresh Token
http://localhost:3000/user/getDetails	GET	Gets the details of user himself and admin can get the details of all users
http://localhost:3000/user/deleteUser?mobileNumber=9876543216	DELETE	Remove the particular user himself and by phone number if admin deletes from the database
http://localhost:3000/user/updateUser?mobileNumber=9876543216	PATCH	Update user details except password and admin could not change user email, phone number.
http://localhost:3000/user/changePassword	PATCH	Update own password of user or admin
http://localhost:3000/user/generateAccessToken	POST	Generate access Token from refresh token when it expires.
http://localhost:3000/user/revoke	POST	To revoke the access and refresh token of logged in User
http://localhost:3000/user/revokeAll	POST	To revoke all the access and refresh token of logged in user.

Architecture



SignUp

In this we are providing all the users details like name, mobile number, email, password, age, country and scope as an optional.

We are providing X-Secret Key. If it is provided correct then it is treated as Admin. If the X-Secret Key is wrong/empty then it is treated as User.

Admin has all the scopes if the scope is not provided in the request body and user has GET, UPDATE, POST. If the scopes are given then the particular scopes are treated.

Password is hashed using scrypt and then stored in the collection.

All the details are stored in the User Collection with UUID.

Login

01 User is asked to provide Email and password in the request Body.

02 Only the required request body key value pairs are allowed. No extra or empty body is allowed.

03 After Checking credentials Access and Refresh Tokens are generated and stored in different Collections based on the roles in UserToken and AdminToken collection.

04 Tokens are generated using sign function with private key, payload and SignInOptions. RSA algorithm is used with roles, scope, UUID in payload.

Update User

In this we are updating the individual details based on checking the role and scope.

We are providing header as Access Token and password needed for updating email, mobile Number of self.

Admin can update other admin or user by giving mobile number in the query params.

Admin cannot update email, mobile number of other admin/user. User can update only his details.

If User updates his email or phone number he has to provide password.

NOTE- Scope is checked for every individu
before updating details.

Delete User

01 In this we are deleting the user from the User collection.

02 User is asked to provide access token in the header.

03 Admin can delete the particular user if he provides the mobile number of the user. If the admin do not provide any params then he can delete himself.

04 User can only delete himself if he has a scope of deleting himself.

Get Details

In this we are getting the individual details based on checking the role and scope.

We are providing header as Access Token and users=”all”, mobile number as params.

Admin can get details of other admin or user by giving mobile number in the query params.

Admin can get all the users and other admins details if he provides users = “all” in the params.

User can only get his details.

NOTE- Scope is checked for every individual before getting details.

Generate Access Token

01 In this we are generating access token from the refresh token.

02 User is asked to provide refresh token in the header.

03 Both admin and user can generate their access token when it expires.

04 After generating the access token it is replaced by the expired access token in the collection.

Change Password

In this we are changing the individual password.

We are providing header as Access Token and old password which is first checked if it matches.

New Password is passed in the request body and it is hashed using scrypt.

Old password is replaced by the new password in the User collection.

Both user and admin are allowed to change password and have rate limiter which allows to change password once in a month.

Revoke and Revoke All

01 In this we are revoking access token and the refresh token in revoke API.

02 User is asked to provide refresh token in the header.

03 In revokeAll API all the access token and refresh token associated with the user/admin are revoked.

04 It is always checked in every API after the login that the tokens are revoked or not. If the tokens are revoked then the boolean value is true in the collection.

Indexing With Unique and TTL

- Indexing in MongoDB refers to creating data structures that enhance the speed of data retrieval operations, such as queries and sorting, by allowing MongoDB to quickly locate and access relevant documents.
- Indexes are created on specific fields or combinations of fields within a collection.
- MongoDB uses B-tree data structures for indexing, which provide efficient search and retrieval capabilities
- **Index Types: Single Field, Compound, Multi-key, geospatial, text index.**
- **Properties :** A unique index ensures that indexed fields have unique values across documents in a collection.
- A TTL index allows MongoDB to automatically remove documents from a collection after a specified period (time-to-live) has elapsed.

Rate Limiter

01 A rate limiter is a mechanism to control the rate of incoming requests or actions from a client or user.

02 It sets limits on how many requests or actions a client can make within a specified timeframe.

03 helping to prevent abuse, maintain system stability, and ensure fair usage of resources.

04 Helps to prevent denial-of-service (DoS) attacks or excessive usage that could degrade system performance. It helps maintain system stability and prevents overload / traffic.

THANK YOU!

ROHIT YADAV